新春寄语 | 35年码龄程序员：人生漫长，走点弯路在所难免

2020-01-24 吴咏炜

现代C++实战30讲 进入课程 >

讲述：吴咏炜
时长 17:49　大小 16.33M

你好，我是吴咏炜。

当你读到这一讲的时候，新年已经要来临了。今天，我们就轻松点，不讲硬核的话题了。我们就一起闲聊一下如何学习编程。

我的编程生涯

我学习编程在我的那一代人里算是比较早的。那时候已经有了“学电脑要从娃娃抓起”[illegible]法，所以初一时老师就给我们介绍了什么是电脑：我第一次见到的是一台 Z80 芯片的 Laser 200，连接在一台单色显示器上，有着集成的键盘，ROM 里装载着 BASIC 语言解释

器，有 4 KB 内存（你没有看错单位），没有磁盘、鼠标这些我们熟悉的外设，但可以连磁带机，还可以做比计算器更复杂的计算工作！

我还记得我尝试的头几个程序里有一个是出十道计算题，用随机数生成题目，让用户输入答案，并在最后计算得分。记得一开始我对 `For … Next` 循环的语义理解还不正确，找了计算机老师帮忙才得到正确的结果。但得到想要的结果后，真是有成就感啊！

于是，我就被这个按键会滴滴响的破烂玩意儿迷上了，还央求爸爸妈妈花了足足 500 元大洋（那时我爸妈的月收入才 60 几元，我妈因此放弃了一条金项链……）给我买了台更先进的 Laser 310（有 18 KB 内存哎！），连在家里的老式电视机上，这直接或间接导致了我的视力在初中急剧下降。不过，我的学习成绩完全没有受到负面影响，还各科全部稳定上升，直到考进了理想的高中和大学。

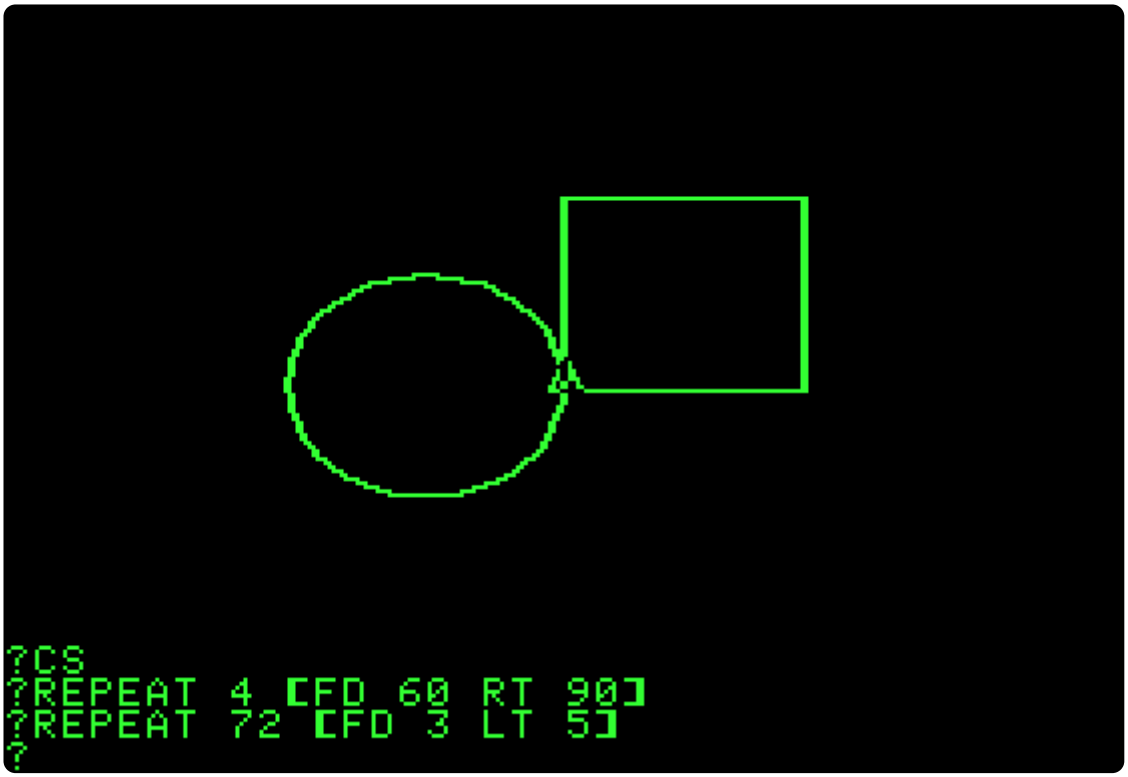
当然，没有磁盘的 Laser 310 实在是太弱了。还好，我参加了少科站的计算机班，逐渐接触到了 Apple II 计算机，后来又接触到最早的 IBM PC，并一路学习了：

> BASIC：当然，BASIC 还是少不了，在我们那个年代，它一直是很主流的，竞赛什么的还是常用 BASIC。Apple BASIC 和 DOS 3.3 就是我高中玩得最多的系统环境了。记得因为没有递归，在 BASIC 里写快速排序还是一件非常痛苦的事……

```
    1000
40  VTAB 4:A$ = "DOS VERSION 3.3
      SYSTEM MASTER": GOSUB 1000

50  VTAB 7:A$ = "JANUARY 1, 1983
    ": GOSUB 1000
60  PRINT D$;"BLOAD LOADER.OBJ0"

70  CALL 4096: REM  FAST LOAD IN
     INTEGER BASIC
80  VTAB 10: CALL  - 958:A$ = "C
    OPYRIGHT APPLE COMPUTER,INC.
     1980,1982": GOSUB 1000
90 C =  PEEK ( - 1101): IF C = 6
    THEN  PRINT : INVERSE :A$ =
    "BE SURE CAPS LOCK IS DOWN":
     GOSUB 1000: NORMAL
100  PRINT  CHR$ (4);"FP"
1000  REM  CENTER STRING A$
1010 B =  INT (20 - ( LEN (A$) /
    2)): IF B =  < 0 THEN B = 1
1020  HTAB B: PRINT A$: RETURN

]
```

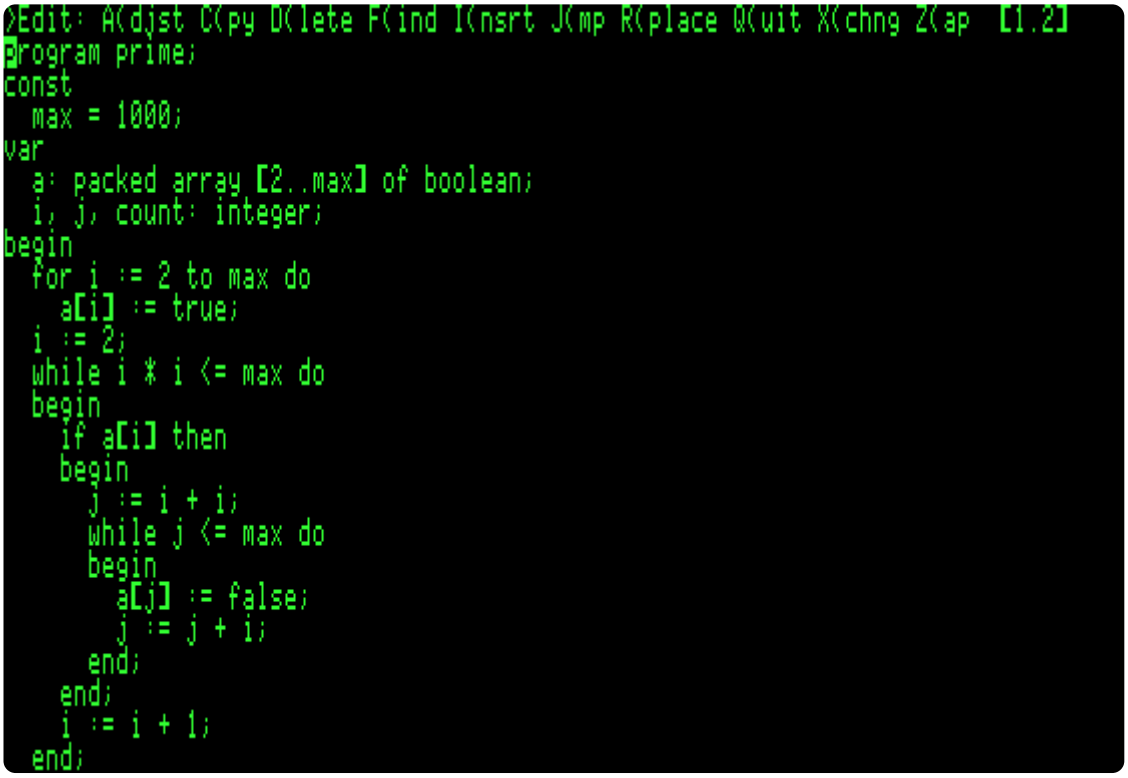
Apple DOS 3.3 的 HELLO 程序列表

Logo：当时曾经热过一阵，但除了提高了我对编程的热情，似乎没记得它对我有多大的帮助。当时就只是玩玩作图而已，没有领会到它背后的编程理念。



Apple Logo 的界面：有没有注意到圆不圆，方也不方？因为硬件像素点就不是正方的……

Pascal：比起 BASIC，绝对是一阵新鲜空气啊。从 Pascal 里，我最早领会到了什么是结构化编程。记得最初需要理解指针就是在 Pascal 里。麻烦的地方是，在 Apple II 上跑 Pascal 没有 80 列卡是十分痛苦的，而少科站只有一台机器有 80 列卡……



UCSD Pascal 的程序编辑界面：这里用上了 80 列卡

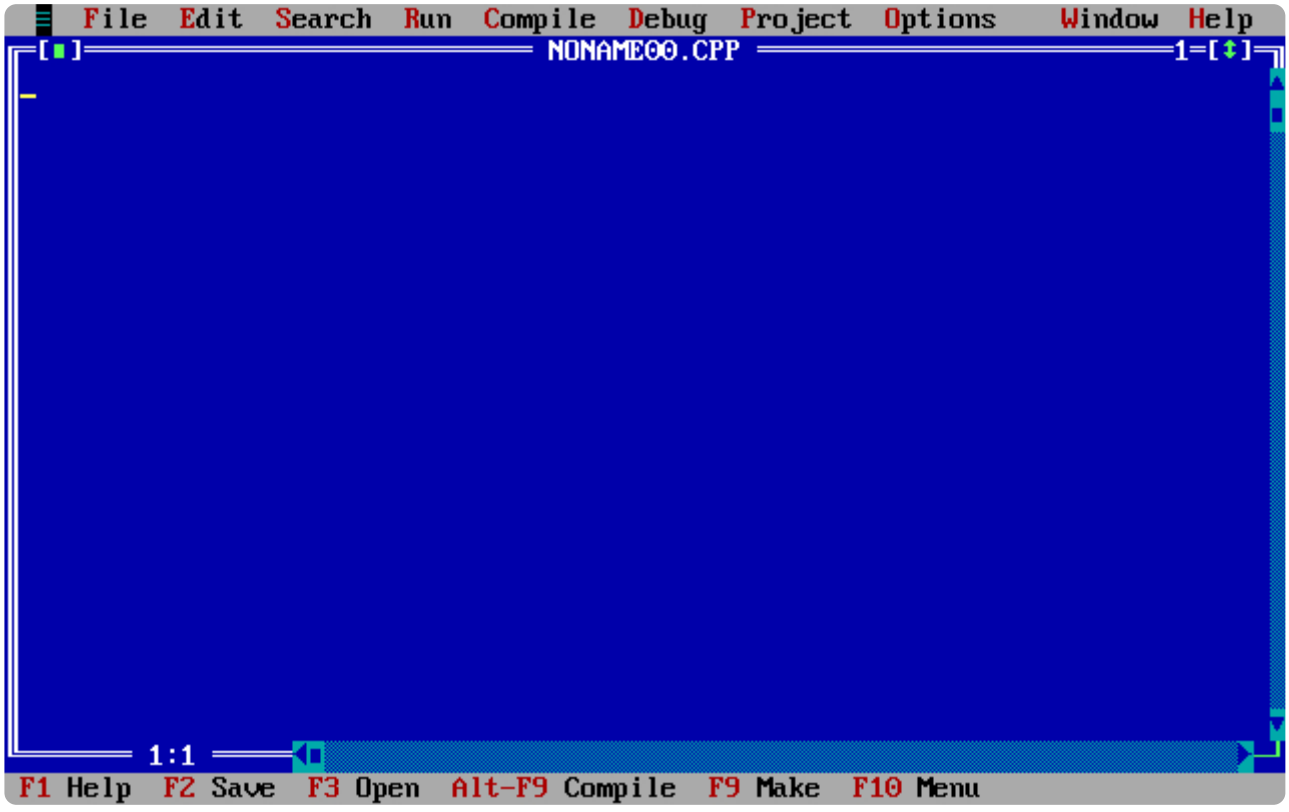
Prolog：最初在纸面上学了点，但对我个人没有产生深远的影响。整个编程生涯里实际运行 Prolog 程序的机会都没有很多次。

Lisp：学 Prolog 的时候也一起了解了。但当时只是了解了语法，而没有理解它的编程范式，也完全没有实践环境，因而没什么用。真正对 Lisp 有所领会要到多年之后捡起 Scheme 和 Racket 的时候。

Forth：有段时间我很迷 Forth，但同样，因为没有实践的机会，实际上影响可以忽略不计。

C：少科站计算机老师大力推荐的语言，但跟 Pascal 一样，跑起来不容易——在 Apple II 上需要插 Z80 卡跑 CP/M 操作系统才行。在 Turbo C/C++ 到来之前，没有太多机会使用。

C++：接触到 Turbo C++ 之后，使用 C++ 就是件顺理成章的事了。具体哪一年开始用，我已经记不清了。印象比较深刻的，是 94—95 年写毕业论文时，一开始用的是没有数学协处理器的 386，作图的时候可以看到点是一个个出来的。后来换了有协处理器的 486 后，点就是一串串地流淌出来的。



网上找来的 Turbo C++ 集成开发环境界面

在我学习计算机的年代，一个高性能的编程语言属于实际应用的刚需，所以 C++ 成了后面非玩具项目的不二选择。大学期间肯定是花了不少时间把玩 C++ 的（依稀记得有个实验室老师希望我们有人接手他的一个 Fortran 计算代码，但没人肯上）。不过，那段时间对 C

和 C++ 的理解还是有缺陷的，再加上 DOS 下没有保护模式，也不那么容易发现错误，C++ 的路不太好走。记得曾经遇到个程序发生了空指针错误，我一直没找到原因……

大学之后，又陆续接触了一些其他的开发语言，如 VBScript 和 JavaScript。而我对 C++ 的理解有真正长足的进步，是在 2000—2005 那段时间。那时候，我在一家网络安全公司工作，对网络上的实时报文做高性能协议分析和内容检测，需要对网络协议、面向对象编程、系统架构都有比较好的把握。这又恰恰是互联网越来越成熟、走进大众生活的年代，网上的开发资源也愈见增多。我那时在 Windows 上开发 Linux 应用程序，为了方便起见，在本机也安装了 GCC，因而有段时间我在 MinGW GCC 的邮件列表还是相当活跃的。因为有切实的工作需要，也因为有好的交流场所，那是我真正在开发技能上突飞猛进的时间。

再后面，我加入了 Intel，并在那里工作了十年多。Intel 是一家了不起的公司，我在那里也学到了很多东西。不过，纯粹从软件开发技能上来说，那段时间进步不多。如果我坚持一直待下去的话，应该会有机会成为 PowerPoint Engineer 的。😂

离开 Intel 后，我的主要开发语言从 C++ 变成了 Python。这也算是一种全新的体验吧。不同的开发语言有着不同的功能取舍，而 Python 放弃了性能，来换取表达能力上的提高，对开发人员来说真是轻松之极。我在好几年的 C++ 大会上，讲 C++ 时都会拿 Python 代码来对标。一开始还有点不服气的意思（我也就比你多写了那么几行嘛），后来成了由衷的佩服和学习。我觉得 C++ 的发展目标就应该是达到 Python 的表达能力，同时还具有高出几个数量级的性能，那样就完美了……

一万小时定律

说了半天，你有没有觉得我学得非常的杂啊？我现在自己想想，都觉得似乎有点太杂了。不过，人生在世，不走点弯路是不可能的。我也没觉得我花在那些小众语言或不重要的特性上的时间完全是浪费，虽说也许有更好的学习方式。学习编程绝对是一场比马拉松更长的长跑，是需要长期的努力和付出的。

你很可能已经听说过“一万小时定律”了。这个说法的核心意思是，在大部分的领域里，从普通人变为专家需要投入至少约一万小时的时间。如果按每天工作八个小时、一周工作五天计算，那么成为一个领域的专家至少需要五年。如果投入时间不那么多，或者问题领域更复杂，“十年磨一剑”也是件很自然的事。

注意这是个约数，也只是必要条件，而非充分条件。时间少了，肯定完全没有希望；大于等于一万小时，也不能保证你一定成为专家。即使天才也需要勤学苦练。而如果天资真的不足，那估计投入再多也不会有啥效果。

如果你看得更仔细一点，你在了解一万小时定律时，应该会注意到不是随便练习一万小时就有用的。你需要的是刻意的练习。换句话说，练习是为了磨练你的思维（或肌肉，或其他需要训练的部分），而不能只是枯燥的练习而已。

打另外一个比方，如果你工作了五年，任务是数据库的查询和报表。那估计对绝大多数人，只有第一年可能是有能力提高的，后面四年则对能力提高没有什么大帮助。在求职市场上，这样的五年经验，多半不比两年工作经验的更有竞争力。考虑到五年经验的的薪水比两年的要高，很多公司会更偏好经验更少的呢。这就是为什么纯编程的职位（像 Java 开发工程师、Python 开发工程师之类）基本上见不到要求十年以上经验的。C++ 这方面相对还略好些，因为大家都知道 C++ 上手需要的时间比较长，所以经验很少的 C++ 程序员意义不大，除非公司里能分配比较有经验的程序员指导。但总体而言，经验要求多的，一般更会是架构、管理方面的职位，大家更认同这些方面的能力更需要长时间的积累。

你有没有注意我说的是“大家更认同”，而不是“我认为”？事实上，我个人的观点恰恰相反，纯粹的程序员能力，不论何种语言，也是可以积累的。只不过，这种积累的有效性因人而异，因而在求职市场上参考意义偏弱些。程序员如果希望积累个人能力的话，很重要的一点，是不要把重复的事情一直重复下去。如果你做一件事感觉以前做过了，你就得想想这里面是否有共性了；如果你做一件事已经是第三第四遍了，那绝对是某个地方出了点问题。你应当做的事情，是退一步，看看如何可以用更好的抽象来解决问题，而不是一遍遍地去重复——否则，就真成了码农，潜在可以被其他人，甚至机器，在某一天替代了。

这点上来说，C++ 是很适合爱好抽象的程序员的语言。C++ 允许高效的抽象，可以用来构造新的领域专用语言。我个人觉得 C++ 挺适合分层开发的——一部分有经验的老手开发系统的框架和底层抽象，并暴露出一个简单易用的接口给上层开发者；上层开发者可以相对经验较少，在老手的指导下完成整个系统的构建。

估计我说了这么多，你还是会想，那你学了这么多乱七八糟的语言，有用吗？

我的个人意见是，既没用，也有用。

如果我有一个好的老师具有针对性地来指导我，指出我哪方面有欠缺，哪方面需要加强训练，那当然会是件极好的事。可那样的事似乎只在音乐界和体育界发生了。在编程这样的开放领域，我还真不知道哪个大师是好老师教出来的。我不是说老师没有用（我遇到过好几个对我编程生涯有影响的好老师），但存在一个一对一的、同领域的好老师的概率真的太低了，尤其在你成长到一定程度之后。编程绝对是一个需要自学的领域。我在编程上“荒废”的时间，应该属于这一万小时中比较低效的部分，但不是无效的部分。说低效，是因为我并不是针对我的编程能力方面的缺陷，去刻意提高自己在某方面的能力。但是，因为我也不是无聊地重复编写同样的程序，而是根据自己的兴趣爱好钻研自己不理解、感兴趣的方面，这种“泛读”也应该部分起到了塑造我的思维的作用。

我还很喜欢 Larry Wall 认为程序员该有的三大美德：懒惰，急切，傲慢（laziness, impatience, hubris；初次阐释于 *Programming Perl* 第二版）。我翻译出完整的原文与你妙文共赏：

> **懒惰**
> 使得你花费极大努力来减少总体能量开销的品质。懒惰使你去写能让别人觉得有用、并减少繁杂工作的程序；你也会用文档描述你的程序，免得你不得不去回答别人的问题。因此，这是程序员的第一大美德。
> **急切**
> 当计算机不能满足你的需求时你所感到的愤怒。这使得你写的程序不仅满足自己的需求，还能预期其他需求。至少努力去这么做。因此，这是程序员的第二大美德。
> **傲慢**
> 老天都受不了你的极度骄傲。这种品质使得你写程序（和维护程序）时不允许别人有机会来说三道四。因此这是程序员的第三大美德。

搞软件开发的，我相信没有很笨的。智力上的差距通过努力、态度和技巧是完全可以填补的。希望你也能拥有这三个懒惰、急切、傲慢的美德。

英语学习

学编程应该学好英语。学编程应该学好英语。学编程应该学好英语。（重要的事情说三遍。）

不管出于什么原因，中文的编程资源没有英文的好，这是不争的事实。前些天，我在帮朋友出些 C++ 的考题，也在网上找了一下，结果看到了很多错误的题目和答案。有还在使用

`void main` 的（应该是 `int main`），有对结构的对齐规则理解不正确的（部分是因为 MSVC 和 GCC 规则有些细微的小区别），还有对 C++ 对象的生命周期规则理解有偏差的，等等。

事实上，我对 C++ 有疑惑时，解答大部分来自于 Stack Overflow——而国内完全没有同等级的编程问答网站。形式上有点相似的是百度问答，但内容的质量——不管是问题，还是回答——完全不可同日而语。百度问答上的很多问题，如果有人胆敢贴到 Stack Overflow 上去，问题立即会变成负分……

怎么学好英语，这个问题是被讨论过无数次了。对于我来讲，非常简单：我喜欢英语。喜欢永远是最好的学习动力。那如果你就是喜欢不起来呢？那我觉得，你需要告诉你自己

1. 学好英语是必要的
2. 学好英语比学好编程容易
3. 还有任何疑问，请参见第 1 条

你需要逼自己一下，定下小小的目标和计划，并每天**坚持**。比如：

- 每天阅读一篇英语的编程文章
- 每天看一条英文的 C++ Core Guideline
- 每天看 5 页 C++ 之父的 *The C++ Programming Language*（或其他的英文编程书籍）
- 每天在 Stack Overflow 上看 3 个问答
- 等等

你很快会发现，克服心理障碍之后，学习英语比起编程来是不费吹灰之力的事，不需要动什么脑筋。把自己沉浸到一个英文环境里，英语能力的提高是件自然而然的事。听说读写，其他方面缺乏训练还情有可原，读是应该最容易被解决的。为了能和世界各地的其他优秀程序员网上交流，写也需要多加练习——不要怕有错误，用简单的词把意思表达清楚，就是很好的第一步。多读多写，基本的网上阅读和交流就不会是一个难题了。

新春快乐

听我唠叨了这么多，希望你没有觉得厌烦。哈哈，就在此祝你新春快乐，在新的一年里学习顺利、工作成功。轻松的生活不是成长的人生所需要的，但希望你遇到的困难都能找到解决的方法，进而有所提高。

在这里顺便告个假，请容我和编辑稍稍歇上一天，1 月 27 日也就是大年初三我们停更一次。大家过年期间除了和亲朋好友团聚外，也可以抽时间把前面的内容再复习下。初五我会为大家提供一个加餐，C++ 编程推荐书目，希望能够为你在这个课程之外进一步提升自己的能力提供参考方向。1 月 31 日起我们恢复正文更新，再上征途，完成最后五讲的学习。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　26 | Easylogging++和spdlog：两个好用的日志库

下一篇　新春福利 | C++好书荐读

精选留言 (7)

写留言

花晨少年
2020-01-24

30出头了，心里也有些犹豫，不知道接下来还能写多少年代码，我想一直写下去，不知道大环境支不支持，都说35如果是道坎，老师有机会分享下经验吧

展开

作者回复: 如果喜欢，自己又擅长，就可以写下去呀。

如果热爱、擅长并不断学习提高，45岁都不是什么坎……

2

三味

2020-01-31

我平时看到带“30岁”这个字眼的文章就炸。我研究生毕业都快三十了，总不能刚毕业就失业吧！从25是个坎，到30岁是个坎，再到35，我想国内就算是互联网，发展也没有二十年吧？戒骄戒躁，戒骄戒躁。

作者回复: 哈哈，不用理那些说法。自己变强是硬道理。

tt

2020-01-27

老师新春快乐！

展开

作者回复: 新春快乐！

哪儿都不能去了。现在我只能闷头写稿，你们可以努力学习和复习。😎

许山山

2020-01-26

老师新年快乐，平安健康！

展开

作者回复: 谢谢谢谢🙏。现在平安健康最好……

LiKui
2020-01-24

祝大家新年快乐，百毒不侵

展开

作者回复: 嗯，新年不出门，闷头学习……😂

Vackine
2020-01-24

吴老师，新年快乐！专栏要反复回看😊

展开

作者回复: 新春快乐！有空多多复习，多多练习。😁

申学晋
2020-01-24

学习英语的建议非常好👍

展开

作者回复: 谢谢。开阔眼界是必需的，语言则是必要的工具。